Python Programming for Statisticians

Zhifeng Wang

Department of Statistics
Florida State University
The Graduate Modern Statistics Club (GMSC) is a student club of Statistics Department, focusing on modern statistics including big data computation, statistical machine learning, high dimensional statistics, modern optimization theory and other exciting research areas.

We hold weekly meetings: Friday 2:00-3:30pm OSB 205

Our webpage: https://stat.fsu.edu/gmsc. (meeting schedule is available online)
This presentation is just a brief introduction, aiming to arouse your interest, or provide you a study guide.

- Basics of Python
- Data Manipulation
- Visualization
- Modeling and Analysis
Intro to Python
Basics
Data Manipulation
Visualization
Modeling and Analysis

Python (1990)

Advantages:

- a high-level interpreted language (open source)
- easy to learn, efficient in coding
- much faster than R, scalable for large-scale applications
- growing number of libraries for statistical analysis
  - NumPy
  - SciPy
  - Pandas
  - Scikit-learn
  - Matplotlib

Drawbacks:

- Slower than compiled languages (e.g., C, C++)
- Hard to encrypt your code
Example: Histograms

Figure: A histogram
Example: Boxplots

Figure: Boxplots
Example: Advanced plots
Example: Image Operation

Figure: Image Operation
Example: Sparse Regression

Figure: Solution Paths in Variable Selection
Example: Decision Tree Regression

Figure: Tree Regression
Example: Decision Tree Classification

Figure: Classification Tree
Example: Support Vector Machine (SVM)

Figure: Support Vector Machine
Example: Clustering

Figure: Comparing different clustering algorithms on toy datasets
Example: Random Forest

**Figure**: Feature Importances
Example: Deep Learning

Figure: Detecting facial keypoints using Deep Learning
Basics of Python
Getting help

>>> help()

Welcome to Python 3.5's help utility!

If this is your first time using Python, you should definitely check out the tutorial on the Internet at http://docs.python.org/3.5/tutorial/.

Enter the name of any module, keyword, or topic to get help on writing Python programs and using Python modules. To quit this help utility and return to the interpreter, just type "quit".

To get a list of available modules, keywords, symbols, or topics, type "modules", "keywords", "symbols", or "topics". Each module also comes with a one-line summary of what it does; to list the modules whose name or summary contain a given string such as "spam", type "modules spam".

help>
Comments

- Use the hash sign for single-line/inline comments.

  ```python
  >>> # This is the first comment.
  >>> print("Hello, world!") # inline comment
  Hello, world!
  ```

- Triple-quoted strings span multiple lines and serve as multi-line comments.

  ```python
  >>> ""
  ... This
  ... code
  ... does
  ... nothing.
  ... ""
  ```
**Core Data Types** (makes programs easy to write):

<table>
<thead>
<tr>
<th>Object Type</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numbers</td>
<td>int(1234), float(3.14), complex(3+4j)</td>
</tr>
<tr>
<td>Strings</td>
<td>'spam', &quot;I'm hungry.&quot;</td>
</tr>
<tr>
<td>Lists</td>
<td>[ 1, &quot;two&quot;, [3, 4] ]</td>
</tr>
<tr>
<td>Tuples</td>
<td>(4, 8, True, &quot;hello&quot;)</td>
</tr>
<tr>
<td>Sets</td>
<td>{4, 8, True, &quot;hello&quot;}</td>
</tr>
<tr>
<td>Dictionaries</td>
<td>{'pi': 3.14, 'e': 2.71}</td>
</tr>
</tbody>
</table>
**Indentation & Statements**

**Indentation:**
- In python, **indentation** is used to denote **nested blocks** of code.
- Indentation must be **consistent** (typically, we use 4 spaces per level).

**Statements:**

<table>
<thead>
<tr>
<th>Statement</th>
<th>Role</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assignment</td>
<td>Creates References</td>
<td><code>i=0; x=[0,1]</code></td>
</tr>
</tbody>
</table>
| if/elif/else| Select an action     | `if temperature < 70:
                       print("Wear long pants.")` |
| for       | Iterate Sequence    | `for ch in 'Hello':
                       print('Next Letter:',ch)`          |
| while     | Loop                | `while temperature < 70:
                       print('Keep Working')`            |
Strings: where Python is strong

- Strings are stored as sequences of characters.

```python
>>> fruit = "orange"
>>> fruit[0], fruit[-2]
('o', 'g')
>>> fruit[1:3], fruit[1:], fruit[:-1]
('ra', 'range', 'orang')
>>> fruit[::2], fruit[::−1]
('oag', 'egnaro')
>>> list(fruit)
['o', 'r', 'a', 'n', 'g', 'e']
```

- You can concatenate strings.

```python
>>> "abc" + "123"
'abc123'
>>> "abc" * 2
'abcabc'
```
Here is a real interview question:

*given an integer, compute the sum of the cubes of its digits.*

In python, we only need **one line:**

```
| sum(int(c)**3 for c in str(num)) |
```
A “new” data type: dictionaries

**Dictionary**

- consists of **unordered** “key + value” pairs (a hash table)
- fast lookups, but **high memory usage**

```python
>>> d = {'pi':3.14, 'e':2.7}
>>> d['pi']  # Query key
3.14
>>> d['zero'] = 0  # Adding/modifying
>>> print(d)
{'e': 2.7, 'zero': 0, 'pi': 3.14}
```
Data Manipulation

(Just a beginning!)
Useful modules

**NumPy**
- $n$-dimensional array
- random number
- basic linear algebra

**SciPy**
- linear algebra
- optimization
- image processing

**Pandas**
- data structures and operations
- time series
Arrays in NumPy

```python
>>> import numpy as np
>>> myarray = np.array(range(6))
>>> myarray
array([0, 1, 2, 3, 4, 5])  # enter a 1-d array
>>> myarray.shape  # check the shape
(6,)
>>> myarray.shape = 3, 2  # reshape myarray
>>> myarray.shape
array([[0, 1],
       [2, 3],
       [4, 5]])
>>> np.array([[0, 1], [2, 3], [4, 5]])  # directly enter a 2-d array
array([[0, 1],
       [2, 3],
       [4, 5]])
```
DataFrame in Pandas

```python
>>> import numpy as np
>>> import pandas as pd
>>> dates = pd.date_range('20160101', periods=6)
>>> dates
DatetimeIndex(['2016-01-01', '2016-01-02', '2016-01-03', '2016-01-04', '2016-01-05', '2016-01-06'], dtype='datetime64[ns]', freq='D')

>>> df = pd.DataFrame(np.random.randn(6,4), index=dates, columns=list('ABCD'))
>>> df
       A         B         C         D
2016-01-01 -0.845099  1.273944  1.223346  0.334451
2016-01-02  0.690085  1.041370  0.765391  1.792459
2016-01-03  0.849334 -0.618640  0.102290  0.463240
2016-01-04  0.432989 -0.972492  1.517998  0.869123
2016-01-05  1.975331 -0.934912  1.366209 -0.475946
2016-01-06 -0.688269 -0.075794  0.528336 -0.559262
```
More things are waiting for you!

- Many methods/functions/attributions are **omitted** here.
- Those rules and syntaxes are **important and useful**.
- What can you do further?

```python
>>> dir(df)
['A',
 'B',
 'C',
 'D',
 'T',
 '_AXIS_ALIASES',
 '_AXIS_IALIASES',
 '_AXIS_LEN',
 '_AXIS_NAMES',
 '_AXIS_NUMBERS',
 '_AXIS_ORDERS',
 '_AXIS_REVERSED',
 '_AXIS_SLICEMAP',
 ......```
Visualization

(Let your results be cool.)
Commonly used modules

- **matplotlib**
  *standard basic data visualization package*

- **seaborn**
  *advanced package for statistical graphs*

- **ggplot**
  *a port by Yhat of the ggplot2 package in R created*
Modeling and Analysis

(Key skills for statisticians.)
# The most popular modules

## Basic Machine Learning Tools: scikit-learn

### Classification
- Identifying to which category an object belongs to.
- **Applications**: Spam detection, Image recognition.
- **Algorithms**: SVM, nearest neighbors, random forest, ...

### Regression
- Predicting a continuous-valued attribute associated with an object.
- **Applications**: Drug response, Stock prices.
- **Algorithms**: SVR, ridge regression, Lasso, ...

### Clustering
- Automatic grouping of similar objects into sets.
- **Applications**: Customer segmentation, Grouping experiment outcomes.
- **Algorithms**: k-Means, spectral clustering, mean-shift, ...

### Dimensionality reduction
- Reducing the number of random variables to consider.
- **Applications**: Visualization, Increased efficiency.
- **Algorithms**: PCA, feature selection, non-negative matrix factorization.

### Model selection
- Comparing, validating and choosing parameters and models.
- **Goal**: Improved accuracy via parameter tuning.
- **Modules**: grid search, cross validation, metrics.

### Preprocessing
- Feature extraction and normalization.
- **Application**: Transforming input data such as text for use with machine learning algorithms.
- **Modules**: preprocessing, feature extraction.

---

**Deep Learning**: Theano, TensorFlow, Pylearn2

*(will be covered by Xin in his presentation)*
Two examples

- Plot different SVM classifiers in the iris dataset.

- Use random forest to obtain the feature importances.
Thank you for coming!

Thank Shao Tang for his Python examples and Xin Sui for his advices.

Deep Learning Tutorial will be given by Xin Sui on Oct. 28 (2-3:30pm). Welcome you to join us!

Don’t forget to visit our webpage: https://stat.fsu.edu/gmsc.